

These are the new changes to the MacBinary Standard, as generally agreed upon in the MacBinary II Conference 6/21/87, and as changed in the followup conference 6/28/87. Revised 7/24/87 to reflect suggestions and clarifications that came later, and to include all necessary information needed from the original MacBinary standard document to implement MacBinary II.

The new standard will be very similar to the original MacBinary standard as described in MACBIN.STD, DL3 of AppDev. (Reading the original standard is recommended for a full understanding of implementation and philosophy behind the MacBinary I and II formats.) The binary format consists of a 128-byte header containing all the information necessary to reproduce the document's directory entry on the receiving Macintosh; followed by the document's Data Fork (if it has one), padded with nulls to a multiple of 128 bytes (if necessary); followed by the document's Resource Fork (again, padded if necessary). The lengths of these forks (either or both of which may be zero) are contained in the header.

The format of the header for MacBinary II is as follows:

Offset 000-Byte, old version number, must be kept at zero for compatibility

Offset 001-Byte, Length of filename (must be in the range 1-63)

Offset 002-1 to 63 chars, filename (only "length" bytes are significant).

Offset 065-Long Word, file type (normally expressed as four characters)

Offset 069-Long Word, file creator (normally expressed as four characters)

Offset 073-Byte, original Finder flags

Bit 7 - Locked

Bit 6 - Invisible

Bit 5 - Bundle

Bit 4 - System

Bit 3 - Bozo

Bit 2 - Busy

Bit 1 - Changed

Bit 0 - Init'd

Offset 074-Byte, zero fill, must be zero for compatibility

Offset 075-Word, file's vertical position within its window

Offset 077-Word, file's horizontal position within its window

Offset 079-Word, file's window or folder ID

Offset 081-Byte, "Protected" flag (in low order bit).

Offset 082-Byte, zero fill, must be zero for compatibility

Offset 083-Long Word, Data Fork length (bytes, zero if no Data Fork).

Offset 087-Long Word, Resource Fork length (bytes, zero if no R.F.).

Offset 091-Long Word, File's creation date

Offset 095-Long Word, File's "last modified" date.

Offset 099-Word, length of Get Info comment to be sent after the resource fork (if implemented, see below).

*Offset 101-Byte, Finder Flags, bits 0-7. (Bits 8-15 are already in byte 73)

*Offset 116-Long Word, Length of total files when packed files are unpacked.

This is only used by programs that pack and unpack on the fly, mimicing a standalone utility such as PackIt. A program that is uploading a single file must zero this location when sending a file. Programs that do not unpack/uncompress files when downloading may ignore this value.

*Offset 120-Word, Length of a secondary header. If this is non-zero, skip this many bytes (rounded up to the next multiple of 128)

This is for future expansion only, when sending files with MacBinary, this word should be zero.

*Offset 122-Byte, Version number of MacBinary II that the uploading program is written for (the

version begins at 129)

*Offset 123-Byte, Minimum MacBinary II version needed to read this file (start this value at 129 129)

*Offset 124-Word, CRC of previous 124 bytes

*This is newly defined for MacBinary II.

All values are stored in normal 68000 order, with Most Significant Byte appearing first then the file. Any bytes in the header not defined above should be set to zero.

The original MacBinary format was ammended to include the sending of the FCMT (Get Info comment) after the resource fork was sent, if the length for such comment, given in offset 99, is not zero. To the best of our knowledge, no program has implemented this feature, due to Apple's stated position that no program should read or write these comments. The definition remains in MacBinary II, so that should Apple ever provide a documented way of reading and writing these comments, terminal programs will be able to take advantage of this feature.

All Finder flags and information would be uploaded, however, a downloading program should clear the Finder flag bits of

- 0 - Set if file/folder is on the desktop (Finder 5.0 and later)
- 1 - bFOwnAppl (used internally)
- 8 - Initd (seen by Finder)
- 9 - Changed (used internally by Finder)
- 10 - Busy (copied from File System busy bit)

Also, fdLocation and fdFldr should be zeroed.

To determine if a header is a valid MacBinary header, check bytes 0 and 74 to be both zero. If they are both zero, either (a) the CRC should match, which means it is a MB II file, or (b) byte 82 is zero, which means it may be a MB I file. (Note that, at the current version level, byte 82 is kept zero to maintain compatibility with MacBinary I. If at some point the MacBinary versions change sufficiently that it is necessary to keep MacBinary I programs from downloading these files, we can change byte 82 to non-zero.)

If the header is a MB II header, the program will check the minimum version byte, to see if it knows enough to decode the file. If the minimum version in the header is greater than the version that the terminal program was written for, it will download the file as pure XModem (creating a "TEXT" file) and notify the user that conversion is needed because the MacBinary version was too high.

If the header does NOT represent a valid MB II header, the program must at minimum check byte 82 to be zero--if it is not zero, the file is not a MB I file. It is possible to write a much more robust routine, by checking the following:

Offsets 101-125, Byte, should all be 0.

Offset 2, Byte, (the length of the file name) should be in the range of 1-63.

Offsets 83 and 87, Long Word, (the length of the forks) should be in the range of 0-\$007F FFFF.

If any of these tests fail, the file is not a valid MacBinary file. It may still be desirable to distinguish between text files and foreign binary files (for stripping line feeds or similar helpful acts). Some tests that would prove useful include: A quantity of bytes in the first block with the high bit set would point to a binary file (though this could be fooled by files with many extended ascii characters, such as generated by the option key on a Mac). A large quantity of zero bytes (nulls) would also point to a binary file.

This proposal was adopted at two conferences attended by representatives from CompuServe, Delphi and BIX networks, and many terminal software publishers. A partial list of those participating is:

Peter Olson/ICONtac
Larry Loeb/BIX
Neil Shapiro/Maug
Mark Hagerman
Michael Pester
William Bond
Bill Steinberg
Don Brown
Bill Davis
Jean Hess
Scott Watson
Clay Maeckel
Harry Chesley
Jack Brindle
Raines/BMUG
Harry Conover
Chris Allen/Dreams of the Phoenix

!! TFORM document
\style mar right 0.6 in;style mar top 0.75 in
\head "[Macintosh Binary Transfer Format ("MacBinary") Standard]"
\foot "- # -|MACBIN.STD"

Dennis F. Brothers - 13 March 1985
(Revision 1 - 10 April 1985 - Miscellaneous clarifications)
(Revision 2 - 12 April 1985 - Corrected reversal of icon position v,h)
(Revision 3 - 6 May 1985 - Added processor ID, general cleanup)

\set justify on
\set join on

The following notes are a proposal for a standard format for binary transfer of arbitrary Macintosh documents via a telecommunication link. It is intended for use both between Macintoshes running (possibly different) terminal programs which adhere to the standard, and for use in uploading arbitrary Macintosh documents to remote systems (where it is presumed that they will be stored as an exact image of the data transmitted). A proposal is also made for standard processing to be performed on text files transferred via a protocol, to maximize the likelihood that text files transmitted to a remote system will be usable by that system, and that text files received from a remote system will be usable by the Macintosh.

It is recommended that the format and procedures described here be referred to as "MacBinary", and that any terminal program implementing this format and procedures be called "MacBinary-Compatible".

The binary format described is independent of the communication protocol used to accomplish the transfer, and assumes only that an 8-bit transparent transfer can be accomplished. Such protocols as Christensen (usually referred to as XMODEM or MODEM7), Kermit, and CompuServe A or B meet this requirement. Because of the proposed standard's MacTerminal/XMODEM heritage, there is a requirement that the transmitted data be padded (with nulls) to a 128-byte boundary at certain points, but this in no way implies that a block-oriented protocol must be used. The basic format proposed is identical to that used by MacTerminal, by Mike Boich and Martin Haeberli, and can be used with a version of MacTerminal that has had a patch applied to "normalize" its implementation of the XMODEM protocol.

In brief, the binary format consists of a 128-byte header containing all the information necessary to reproduce the document's directory entry on the receiving Macintosh; followed by the document's Data Fork (if it has one), padded with nulls to a multiple of 128 bytes (if necessary); followed by the document's Resource Fork (again, padded if necessary). The lengths of these forks (either or both of which may be zero) are contained in the header.

The format of the 128-byte header is as follows (offsets and lengths are given in decimal):

\set justify off
\set join off
Offset Length Contents

000 1 Zero. This is a "version" byte.
 001 1 Length of filename.
 002 63 Filename (only "length" bytes are significant).
 (the following 16 bytes are a standard Finder Info record)
 065 4 File type.
 069 4 File creator.
 073 1 Finder flags:
 Bit 7 - Locked.
 Bit 6 - Invisible.
 Bit 5 - Bundle.
 Bit 4 - System.
 Bit 3 - Bozo.
 Bit 2 - Busy.
 Bit 1 - Changed.
 Bit 0 - Init'd.
 074 1 Zero.
 075 2 File's vertical position within its window.
 077 2 File's horizontal position within its window.
 079 2 File's window or folder ID.
 (End of Finder Info)
 081 1 "Protected" flag (in low order bit).
 082 1 Zero.
 083 4 Data Fork length (bytes, zero if no Data Fork).
 087 4 Resource Fork length (bytes, zero if no R.F.).
 091 4 File's creation date.
 095 4 File's "last modified" date.
 099 27 Zero fill (reserved for expansion of standard).
 126 2 Reserved for computer type and OS ID
 (this field will be zero for the current Macintosh).

\set justify on

\set join on

Note that it is the responsibility of the receiving terminal program to resolve file name conflicts, generally by somehow modifying the name of received file if there already exists a file with the original name on the target volume.

Note also that, while the original window or folder ID and position may be transmitted, the receiving terminal program would not normally set these items for the received file, but would instead accept the values that the File Manager assigns when it creates the new file.

It is suggested that Macintosh terminal programs implement two modes of protocol transfer: text and document. Text mode is used for unformatted files of type TEXT (with only a data fork), and document mode (using the binary format proposed here) is used for all other files. Document mode may also be used on text files, of course, if it is desired to preserve such things as the file's creator ID or creation date.

The intent of text mode is to provide compatibility with text files on other systems. Toward that end, it is recommended that a linefeed be inserted after each return character as the text file is transmitted, and that, in the case of block-oriented protocols, the last block be explicitly padded with nulls if the text does not end on a block boundary. When receiving in text mode, linefeeds and trailing nulls should be stripped. If a CTRL-Z (Hex 1A) character

is received following all other text (and before any null padding), it should also be stripped (Ctrl-Z is a common text terminator on CP/M and some other systems). Note that the above discussion applies only to text files transferred under some protocol, where an exact image of the transmitted data will be stored in a file on the remote system.

When receiving a file via a protocol, a terminal program may distinguish between text and document modes by examining bytes 0, 74, and 82 of the first 128 bytes received. If they are each zero (and at least 128 bytes have been received), then it is a safe assumption that a MacBinary-formatted document is being received. Terminal programs implementing possible future versions of the proposed standard would, of course, accept an appropriate set of version numbers in byte 0. Note also that compatible extensions of Version 0 of the proposed standard are possible (one such is suggested below) that involve transmission of additional information following the information described here. For this reason, a terminal program should be implemented so as to perform the proper receive procedures for all data sent to it, but to ignore any data that it does not know what to do with.

Since a text-mode document does not contain a file name, it is suggested that when text-mode is detected, a file be opened under a dummy name on the receiving Macintosh, the text written to that file, and the file renamed to a name selected by the user (via an SFPutFile box) after the reception is completed. This will avoid problems caused by indeterminate delays for name selection at the beginning of the file transfer.

It is desirable to allow the user to specify the destination volume in advance of the actual start of transfer for either type of transfer. Two methods are suggested for this: provide a "Select Destination Volume" menu selection, presumably in the menu containing the "Receive File" selection; or query the user immediately after the "Receive File" menu selection is made. Either or both of these methods could be implemented in a given terminal program - the independent "Select Receive Volume" method is particularly desirable if the ESC-a/ESC-b automatic receive facility (see below) is implemented. The volume selection procedure should provide the same functions as the volume selection portion of the SFGGetFile and SFPutFile dialog boxes.

\set justify off

\set join off

First proposed extension to the proposed standard:

\set justify on

\set join on

It is proposed that the binary format described above be extended to allow the transmission of descriptive text with a Macintosh document (specifically, the descriptive text from the Finder's "Get Info" box for the file being transferred). This is to be accomplished in a transparent manner by assigning bytes 99 and 100 of the header described above to be used to hold the length of the descriptive text (or zero, if there is none). The descriptive text, if any, will begin on the 128-boundary immediately following the Resource Fork, if present, else the Data Fork, if present, else immediately following the header if neither fork is present. It is hoped that methods for reading and setting a file's "Get Info" text will be made public at some point.

\set justify off

\set join off

Notes on MacTerminal's XMODEM implementation, and a proposed extension:

\set justify on

\set join on

Familiarity with the Christensen (XMODEM) protocol is assumed in the following discussion.

When doing "Mac-to-Mac" transfers, using the binary format described above, unmodified MacTerminal does not use a true XMODEM protocol, and is therefore incompatible with most non-Mac systems. The differences lie in two specifics: the transmitting Macintosh initiates the transfer by sending the two characters ESCAPE (hex 1B) and "a" (hex 61); the receiving Macintosh is expected to reply with the character ACK (hex 06). The transfer then proceeds using normal XMODEM procedures, except that each of the header and the two forks (if present) is treated as a separate XMODEM transfer, with the transmitting Macintosh waiting for a NAK (hex 15), then sending the blocks of that phase (beginning with a block number of one), then sending EOT (hex 04) and waiting for an ACK (hex 06) from the receiving Macintosh.

It is proposed that a modified procedure be accepted as a standard, to be implemented instead of or in addition to the above-described MacTerminal "Mac-to-Mac" protocol in complying terminal programs. The modified procedure, which is compatible with standard XMODEM implementations, functions as follows: The transmitting Macintosh sends the two characters ESCAPE (hex 1B) and "b" (hex 62). The receiving Macintosh may optionally reply with ACK (hex 06) (this is allowed so as to have minimum impact on existing MacTerminal-compatible implementations). The transmitting Macintosh then awaits receipt of a NAK (hex 15) (or optionally a "C" (hex 43), if the receiving terminal program supports CRC checking), following which a single, normal XMODEM transfer occurs. The transfer may be in text mode or document mode, will begin with block number one, and block numbers will increment uniformly (modulo 256) throughout the transfer.

It is expected that a patch to MacTerminal making it compatible with the above proposed procedure will be available in the near future.

\set justify off

\set join off

Responses to proposals:

Please address comments or questions on the above proposals to:

Dennis F. Brothers
197 Old Connecticut Path
Wayland, MA 01778
617-358-2863

CompuServe: 70065,172

Delphi: DBROTHERS
MCI Mail: DBROTHERS

MacBinary Working Group:

\set justify on

\set join on

An informal working group, consisting of Macintosh terminal program developers and others with interests or expertise in the field of computer communications, was formed during April, 1985 to discuss and refine this proposal. The group met in the MAUG (Micro-networked Apple User's Group) Special Interest Group on the CompuServe Information Service. The present form of the MacBinary format standard represents a consensus of this group as a whole, but may not reflect the opinion of a given individual member of the group.

\set justify off

\set join off

The working group included:

Christopher Allen
William Bond
Steve Brecher
Dennis Brothers
Ward Christensen
Dan Cochran
Mike Cohen
Bill Cook
Ed Edell
Duane Harris
Yves Lempereur
Neil Shapiro
Dan Smith
Bill Steinberg
Scott Watson

\new

\head "[Macintosh Binary Transfer Format II ("MacBinary II") Standard]"

\foot "[- # -]MACBN2.STD"

\set justify on

\set join on

These are the new changes to the MacBinary Standard, as generally agreed conference 6/28/87. Revised 7/24/87 to reflect suggestions and clarifications that came later, and to include all necessary information needed from the original MacBinary standard document to implement MacBinary II.

The new standard will be very similar to the original MacBinary standard as described in MACBIN.STD, DL3 of AppDev. (Reading the original standard is recommended for a full understanding of implementation and philosophy behind the MacBinary I and II formats.) The binary format consists of a 128-byte header containing all the information necessary to reproduce the document's directory entry on the receiving Macintosh; followed by the document's Data Fork (if it has one), padded with nulls to a multiple of

128 bytes (if necessary); followed by the document's Resource Fork (again, padded if necessary). The lengths of these forks (either or both of which may be zero) are contained in the header.

The format of the header for MacBinary II is as follows:

\set justify off

\set join off

Offset 000-Byte, old version number, must be kept at zero for compatibility

Offset 001-Byte, Length of filename (must be in the range 1-63)

Offset 002-1 to 63 chars, filename (only "length" bytes are significant).

Offset 065-Long Word, file type (normally expressed as four characters)

Offset 069-Long Word, file creator (normally expressed as four characters)

Offset 073-Byte, original Finder flags

Bit 7 - Locked.

Bit 6 - Invisible.

Bit 5 - Bundle.

Bit 4 - System.

Bit 3 - Bozo.

Bit 2 - Busy.

Bit 1 - Changed.

Bit 0 - Init'd.

Offset 074-Byte, zero fill, must be zero for compatibility

Offset 075-Word, file's vertical position within its window.

Offset 077-Word, file's horizontal position within its window.

Offset 079-Word, file's window or folder ID.

Offset 081-Byte, "Protected" flag (in low order bit).

Offset 082-Byte, zero fill, must be zero for compatibility

Offset 083-Long Word, Data Fork length (bytes, zero if no Data Fork).

Offset 087-Long Word, Resource Fork length (bytes, zero if no R.F.).

Offset 091-Long Word, File's creation date

Offset 095-Long Word, File's "last modified" date.

Offset 099-Word, length of Get Info comment to be sent after the resource fork (if implemented, see below).

*Offset 101-Byte, Finder Flags, bits 0-7. (Bits 8-15 are already in byte 73)

*Offset 116-Long Word, Length of total files when packed files are unpacked. This is only used by programs that pack and unpack on the fly, mimicking a standalone utility such as PackIt. A program that is uploading a single file must zero this location when sending a file. Programs that do not unpack/uncompress files when downloading may ignore this value.

*Offset 120-Word, Length of a secondary header. If this is non-zero, skip this many bytes (rounded up to the next multiple of 128). This is for future expansion only. When sending files with MacBinary, this word should be zero.

*Offset 122-Byte, Version number of MacBinary II that the uploading program is written for (the version begins at 129).

*Offset 123-Byte, Minimum MacBinary II version needed to read this file (start this value at 129 129)

*Offset 124-Word, CRC of previous 124 bytes

*This is newly defined for MacBinary II.

\set justify on

\set join on

All values are stored in normal 68000 order, with Most Significant Byte appearing first then the file. Any bytes in the header not defined above should be set to zero.

The original MacBinary format was amended to include the sending of the FCMT (Get Info comment) after the resource fork was sent, if the length for such comment, given in offset 99, is not zero. To the best of our knowledge, no program has implemented this feature, due to Apple's stated position that no program should read or write these comments. The definition remains in MacBinary II, so that should Apple ever provide a documented way of reading and writing these comments, terminal programs will be able to take advantage of this feature.

All Finder flags and information would be uploaded, however, a downloading program should clear the Finder flag bits of

\set justify off

\set join off

0 - Set if file/folder is on the desktop (Finder 5.0 and later)

1 - bFOwnAppl (used internally)

8 - Init'd (seen by Finder)

9 - Changed (used internally by Finder)

10 - Busy (copied from File System busy bit)

Also, fdLocation and fdFldr should be zeroed.

\set justify on

\set join on

To determine if a header is a valid MacBinary header, check bytes 0 and 74 to be both zero. If they are both zero, either (a) the CRC should match, which means it is a MB II file, or (b) byte 82 is zero, which means it may be a MB I file. (Note that, at the current version level, byte 82 is kept zero to maintain compatibility with MacBinary I. If at some point the MacBinary versions change sufficiently that it is necessary to keep MacBinary I programs from downloading these files, we can change byte 82 to non-zero.)

If the header is a MB II header, the program will check the minimum version byte, to see if it knows enough to decode the file. If the minimum version in the header is greater than the version that the terminal program was written for, it will download the file as pure XModem (creating a "TEXT" file) and notify the user that conversion is needed because the MacBinary version was too high.

If the header does NOT represent a valid MB II header, the program must at minimum check byte 82 to be zero--if it is not zero, the file is not a MB I file. It is possible to write a much more robust routine, by checking the following:

\set justify off

\set join off

Offsets 101-125, Byte, should all be 0.

Offset 2, Byte, (the length of the file name) should be in the range of 1-63.

Offsets 83 and 87, Long Word, (the length of the forks) should be in the range of 0-\$007F FFFF.

\set justify on

\set join on

If any of these tests fail, the file is not a valid MacBinary file. It may still be desirable to distinguish between text files and foreign binary files (for stripping line feeds or similar helpful acts). Some tests that would prove useful include:

A quantity of bytes in the first block with the high bit set would point to a binary file (though this could be fooled by files with many extended ascii characters, such as generated by the option key on a Mac).

A large quantity of zero bytes (nulls) would also point to a binary file.

This proposal was adopted at two conferences attended by representatives from CompuServe, Delphi and BIX networks, and many terminal software publishers. A partial list of those participating is:

\set justify off

\set join off

Peter Olson/ICONtac

Larry Loeb/BIX

Neil Shapiro/Maug

Mark Hagerman

Michael Pester

William Bond

Bill Steinberg

Don Brown

Bill Davis

Jean Hess

Scott Watson

Clay Maeckel

Harry Chesley

Jack Brindle

Raines/BMUG

Harry Conover

Chris Allen/Dreams of the Phoenix